NASA Technical Memorandum 101710

A Knowledge-Based Flight Status Monitor for Real-Time Application in Digital Avionics Systems

E.L. Duke, J.D. Disbrow, and G.F. Butler

October 1989

National Aeronautics and
Space Administration

NASA Technical Memorandum 101710

A Knowledge-Based Flight Status Monitor for Real-Time Application in Digital Avionics Systems

E.L. Duke
Ames Research Center, Dryden Flight Research Facility, Edwards, California

J.D. Disbrow
PRC Systems Services, Inc., Edwards, California

G.F. Butler
Royal Aerospace Establishment, Farnborough, United Kingdom

1989

NASA
National Aeronautics and
Space Administration
Ames Research Center
Dryden Flight Research Facility
Edwards, California 93523-5000

A KNOWLEDGE-BASED FLIGHT STATUS MONITOR FOR REAL-TIME APPLICATION IN DIGITAL AVIONICS SYSTEMS

E.L. Duke,* J.D. Disbrow,** and G.F. Butler†

ABSTRACT

The Dryden Flight Research Facility of the National Aeronautics and Space Administration (NASA) Ames Research Center (Ames-Dryden) is the principal NASA facility for the flight testing and evaluation of new and complex avionics systems. To aid in the interpretation of system health and status data, a knowledge-based flight status monitor has been designed. The monitor is designed to use fault indicators from the onboard system which are telemetered to the ground and processed by a rule-based model of the aircraft failure management system to give timely advice and recommendations in the mission control room. One of the important constraints on the flight status monitor is the need to operate in real time, and to pursue this aspect, a joint research activity between NASA Ames-Dryden and the Royal Aerospace Establishment (RAE) on real-time knowledge-based systems has been established. Under this agreement, the original LISP knowledge base for the flight status monitor has been reimplemented using the intelligent knowledge-based system toolkit, MUSE, which was developed under RAE sponsorship. In this paper, details of the flight status monitor and the MUSE implementation are presented.

NOMENCLATURE

AFTI	advanced fighter technology integration aircraft
AI	artificial intelligence
Ames-Dryden	Dryden Flight Research Facility of the NASA Ames Research Center
AR	analog reversion mode
ASFM	aircraft sensor and failure management
BCS	backward-chaining system
BNF	Backus-Naur form
FCS	flight control system
FPS	forward production system
FSM	flight status monitor
FSMKB	flight status monitor knowledge base
HiMAT	highly maneuverable aircraft technology
KAT	knowledge acquisition tool
KB	knowledge base
LEX	lexical analyzer generator (standard UNIX program for lexical processing of character input streams)
Mips	millions of instructions per sec
MUSE	name of intelligent knowledge-based system toolkit developed by Cambridge Consultants Ltd. under RAE sponsorship
nb	notice board (blackboard)
RAE	Royal Aerospace Establishment
YACC	Yet Another Compiler-Compiler (standard UNIX program for translating input languages to computer programs)

1 INTRODUCTION

As part of a United States/United Kingdom cooperative aeronautical research program, a joint activity between the Dryden Flight Research Facility of the NASA Ames Research Center (Ames-Dryden) and the Royal Aerospace

*NASA Ames Research Center, Dryden Flight Research Facility, Edwards, CA, USA
**PRC Systems Services, Edwards, CA, USA
†Royal Aerospace Establishment, Farnborough, UK

Establishment (RAE) on knowledge-based systems has been established. Under the agreement, a flight status monitor knowledge base (FSMKB) developed at Ames-Dryden has been implemented using the real-time artificial intelligence (AI) toolkit, MUSE, which was developed in the United Kingdom under RAE sponsorship. Both NASA and RAE have a continuing interest in the application of expert or knowledge-based systems to the monitoring of advanced avionic systems, and this joint program offers benefits to both. Although RAE has been able to evaluate MUSE with a significantly larger knowledge base than those used in earlier tests, NASA now has access to an alternative fast implementation of the flight status monitor (FSM). This cooperation offers wider development opportunities for the future.

To aid in the interpretation of system health and status data, an expert system FSM has been designed. The NASA Ames-Dryden work to produce an expert system for monitoring the status of avionics and digital flight control systems (DFCSs) during flight test is described in Disbrow et al. (1), Duke and Regenie (2), and Regenie (3). During flight testing, problem conditions may be indicated which the test engineer needs to resolve in real time. Such problems can lead to an aborted flight in cases where it may later be determined that the severity of the failure was overestimated or the data misinterpreted because of time pressure. The FSM is designed to use fault indicators from the onboard systems. These indicators are telemetered to the ground and processed by a rule-based model of the aircraft failure management system to give timely advice and recommendations in the control room. A specially developed knowledge acquisition tool (KAT) is used to collect and format the knowledge base which encapsulates expertise on the operation of the onboard systems. Currently, NASA's FSMKB contains over 400 rules which have been implemented in common LISP. This has served as a non-real-time demonstrator of the FSM concept. Several methods for improving the performance of the system to allow effective operation in real time have been (Duke et al. (4)), and continue to be, explored.

Over the last 5 years, there has been a continuing research effort at RAE into the reduction of pilot workload by the use of expert or knowledge-based systems. One of the results of this activity is MUSE, a toolkit designed for the development of real-time AI applications. Cambridge Consultants Ltd. originally developed MUSE under contract to RAE. (An extended version of the package is available commercially.) The toolkit comprises a collection of knowledge representation languages coupled with a set of supporting tools for creating, testing, and delivering AI systems. It provides not only a wide choice of representation techniques, but also the module structuring needed to safely construct major systems. The primary aim of MUSE is real-time and embedded applications characterized by the need to respond to external asynchronous events that interrupt the normal processing flow.

In this paper the implementation of the FSMKB using MUSE is described. Sections 2 and 3 give an overview of the FSM and MUSE, respectively, while section 4 is concerned with the development of the prototype MUSE FSM. Performance and other implementation issues are discussed in section 5.

2 THE FLIGHT STATUS MONITOR

An expert system capable of monitoring the health and status of flight-crucial control systems on high performance research aircraft was developed at the Dryden Flight Research Facility of NASA's Ames Research Center. The goal of the project was to produce a system capable of using an online, real-time application (shown in figure 1) by accepting telemetry downlink data from the aircraft and applying various inference mechanisms to deduce conclusions of interest.

A major concern in advanced high-performance aircraft is the DFCS. These advanced aircraft are often substantially unstable and require augmentation from a full-time, full-authority DFCS. Because these DFCSs are flight critical, the monitoring of such systems is essential in the flight research environment. Problems occurring in the flight control system (FCS) can cause an aircraft to be lost or a flight to be aborted or canceled, or they can force flight test modification. Fast and informative displays relating the health and status of the FCS can save a flight, a mission, or the aircraft itself. Current flight test monitoring technology involves discrete data transmitted from the aircraft and displayed on cathode-ray tubes or light panels with little, if any, interpretation (Fig. 2).

This expert system FSM would process the large amounts of health and status data typical of current DFCSs. An FCS representative of current DFCSs was chosen for analysis and study (Fig. 3), a choice based on experience with various aircraft. This representative control system has many of the characteristics assumed in the development of the FSM and KAT. The FSMKB is based on this representative flight control system.

A conceptual view of the envisioned expert system FSM is shown in figure 1. The system would receive data on the status and health of the FCS from telemetry downlink data and translate the data into a usable format. Conventional programs have been developed that receive telemetered downlinked data and rapidly translate it into digital words. This data is put into the expert system which determines whether any changes have occurred compared with the previous sample and, if so, evaluates the effect of the changes. A data-driven foreground loop determines the state

of the system and informs the user, in this case the systems engineer, of the changes and consequences. If a failure occurs, a warning or caution is issued along with corrective or emergency procedures, when required. As a part of this evaluation, the expert system may ask the user questions on the state of the aircraft. A background task allows the user to query the monitor for information on the FCS state or the rationale used to reach its conclusions. The expert system interrupts the background task, when necessary, to evaluate new data, but if the state of the FCS has not changed between inputs, the expert system does not reevaluate that state.

The knowledge base (KB) contains both aircraft-specific rules as well as metarules, the rules that the systems engineer uses to determine the correct action for a failure situation. The FSM uses these rules to model the failure detection system of the FCS and compare the state generated with that of the aircraft's state. If the monitor's conclusions disagree with the aircraft state, a warning is issued and the user can ask the system to resolve the conflict. The conflict resolution is processed as part of the background task so as not to interfere with the higher priority task of evaluating the aircraft data as it is received.

2.1 Structure of Flight Status Monitor

The FSM consists of several separate expert systems, each with its own inference mechanism. The internal structure of the FSM is shown in figure 4. These inference mechanisms are predominantly forward-chaining, data-driven processes. The aircraft sensor and failure management (ASFM) expert system uses a forward-chaining mechanism to model the aircraft failure management system and to deduce conditions of concern or danger based on the failure indicator information. A metamonitor expert system deduces situations of concern based on knowledge of deductions from the ASFM expert system and the aircraft failure management system. The situations of concern deduced by the metamonitor are analyzed by a fault isolation expert system that deduces probable causes of conflicts, recommends corrective actions, and issues warnings. These expert systems provide detailed system status information and perform a function comparable to that of a flight systems expert.

The system operability expert system uses knowledge of the system effectiveness and the detailed system status data to provide a high-level assessment of the FCS's ability to control the aircraft, complete a specific mission, or function in a given mode. This assessment is performed by a backward-chaining mechanism using hypotheses in an order established by the user. The order of the hypotheses is important because it provides a means for the expert system to determine priorities; the system uses this knowledge of priorities to determine the highest level at which the system is operable and provides this information to the user. The system operability rules are also used to establish the worst consequences of any additional failure.

2.2 Expert System Rules

The rules used in the FSM serve to characterize the FCS of a redundant digital fly-by-wire vehicle. This characterization includes definitions of the FCS health and status data, redundant system elements, emergency procedures associated with FCS failures, and a model of the vehicle's failure management system. The FSM uses several different representations of rules (Fig. 4). Some of these representations are in the form of traditional if-then production rules. However, many of the rules are defined in unusual formats to facilitate definition of the knowledge base and to increase execution speed of the inference mechanisms (Duke and Regenie (2)).

3 THE MUSE INTELLIGENT KNOWLEDGE-BASED SYSTEM TOOLKIT

The MUSE system is a toolkit for the development of real-time applications of AI (Reynolds (5)). It comprises a package of knowledge representation languages, coupled with a set of supporting tools for creating, testing, and delivering AI applications. Also included in MUSE are features suited to real-time operations, such as agenda-based priority scheduling, interrupt handling, and fast data capture. The development environment for MUSE is the Sun workstation, but it can also be tailored to deliver prototype systems on compact solid-state hardware.

3.1 Knowledge Representation Languages

The heart of the MUSE system is an integrated package of languages for knowledge representation (Fig. 5). These languages share the same set of database and object structures, thus allowing them to be freely mixed within a given application. PopTalk, which provides the central component of the package, is a version of the Pop AI language which has been extended to include a complete frame system. It is a block-structured procedural language which supports symbolic processing and therefore much of the AI programming style used in languages such as LISP.

It also provides an object-oriented programming environment in the style of such languages as Smalltalk and is implemented in C for compactness and portability.

Two rule-based languages are provided within MUSE. The first is called the FPS (forward production system), where each rule consists of one or more antecedents followed by one or more consequents. Rule firing is governed by pattern-matches on objects in the MUSE databases, and the rule actions may create or modify database objects or may in turn call on an embedded procedural code to perform special actions. Pattern matching is performed by a modified version of the Rete algorithm used by many production rule systems (Forgy (6)). The second rule-based language is the BCS (backward-chaining system). Like the FPS, it can pattern match on objects in the databases and, as a result of matching, can modify or create objects, or call procedural code. Unlike the FPS, however, the BCS rules can perform searching using a depth-first backtracking strategy similar to that used in Prolog.

3.2 Architecture

The second major component of the package is a set of architectural support facilities. With many AI toolkits the user is restricted to working with a single set of rules and a single database. A key aim of MUSE, however, is to permit the user to partition the application of a collection of well-defined modules. Depending on the nature of the problem to be tackled, a MUSE application can be structured as anything from a single production rule system up to a complex network of cooperating knowledge sources with both shared and private databases. Also included is a powerful data capture system of virtual channels, which allows access to either real or simulated data streams external to the MUSE process.

The module structures are implemented by using the support for object-oriented programming which is built into the PopTalk language, and the basic structure is called a knowledge source. As shown in figure 6, a standard knowledge source comprises a collection of one or more rule sets, each with its own private local database. Separate rule sets are linked by access to a shared database, often referred to as a notice board (nb), which is visible to each rule set in the knowledge source.

A given MUSE application consists of one or more knowledge sources linked by shared access to databases. The control of the scheduling of these separate knowledge sources is carried out by a priority-based "agenda," which maintains an ordered list of tasks to run. The tasks can be scheduled explicitly by rules or procedural code, or implicitly by monitoring changes in databases.

4 IMPLEMENTATION OF THE FLIGHT STATUS MONITOR IN MUSE

The original FSMKB is written in LISP and is not immediately suitable to use as input to the MUSE system. The potential for transcription errors was considerable because of the large number of rules, and it was decided to translate the rules automatically. This process is described in more detail in subsection 4.1. The basic structure of the LISP FSM system has mapped across successfully to the MUSE implementation, although it has proved necessary to include additional rules to perform the voting function. The rules and data have been segmented, where possible, along the lines of the original structure.

4.1 Rule Translation

A set of translation programs was used to convert the LISP into MUSE format. It has the advantages of (1) enabling new releases of the LISP knowledge base to be easily converted to run under MUSE, (2) eliminating the certainty of transcription errors and the time spent locating them, and (3) reducing the translation time. Compiler techniques and formal grammars were used to generate a parser for the original LISP code by way of the standard UNIX programs YACC (Yet Another Compiler-Compiler) and LEX (lexical analyzer generator). The original LISP code was examined and a specification for a lexical analyzer was determined. This was given to the LEX lexical analyzer generator that produced the appropriate program in the C programming language. The original code was then further examined, the underlying grammar was specified in the modified Backus-Naur (BNF) form used by YACC, and the actions to be taken at various stages of the parsing procedure were added. This BNF specification was given to YACC and the C code of a parser for the LISP rules was generated automatically. The combined lexical analyzer and parser were then compiled with a small amount of supporting C code. In operation, the resulting program is fed with the appropriate section of the LISP ruleset and generates MUSE code in a format that can be spliced into a MUSE-structured editor file.

4.2 Structure of MUSE Flight Status Monitor

Each item of data in the MUSE system is represented by an object. These objects are created once on initialization rather than created and destroyed dynamically, as this could cause a large performance overhead due to garbage collection. The MUSE objects for the FSMKB are divided into four object groups: basic, intra, inter, and FCS.

Basic objects represent the failure and status bits and provide the system with the base-level data on which to initiate the reasoning process. These are triplicated to correspond with the three processing channels in the aircraft. Intra and inter objects represent the partial stages of reasoning. The intra objects are associated with the intra channel rules and are also triplicated. The inter objects are associated with the inter channel rules and are not triplicated, since they contain merged information from all three input channels. The FCS objects represent the output of the FCS operability rules; that is, they hold the top-level system status information.

Each object has a name slot which holds an equivalent to the name used in the LISP KB; for example, "Qbar > 413 lb/in.^{2}" The basic objects also have a MUSE system name to examine them in the MUSE browser and to follow updates through the system; these names are generated using the bit number and channel; for example, s49a, f57c. The remaining objects, whose names are not used explicitly, are generated arbitrary system names within MUSE. An object also has a value slot which is used to indicate whether the data item has been set for on or off. For multiple element rules this contains the number of indicators which are on, and in the case of FCS objects, negative values are used to ensure uniqueness.

Communication between rule sets is managed by way of nb's. Four nb's are accessed: the basic_nb, the intra_nb, the inter_nb, and the fcs_nb, corresponding to the object groups previously defined.

4.3 Comparison of LISP and MUSE Rules

The basic structure of the translated rules was designed to mirror as closely as possible the original LISP rules, as shown in figure 7. The following points should be noted:

1. The automatic translation process inserts 'a_' at the start of each item name to ensure that the name commences with an alphabetic character, as required by MUSE.
2. Values 1 and 0 are used in the MUSE code to indicate on and off, respectively.
3. In the MUSE version, the variables beginning with 'I' are "handles" which are used to reference the particular consequents to be changed. This is necessary as items of data are being reused, rather than continuously created and destroyed.

4.4 Rulesets Within MUSE

The rulesets in MUSE correspond to those in the original LISP KB with the addition of the intra_combine ruleset which performs a voting function. These rulesets and their relationships to the nb's are described as follows:

The intra channel rules communicate with the basic_nb and intra_nb notice boards. The incoming data updates are taken from the basic_nb, are reasoned on by the intra channel rules, and the consequents are then placed in the intra_nb notice board. There are potentially three duplicate rule firings, as there are three data channels in the system. This is reflected in the triplicated data in the basic_nb and intra_nb.

The intra multichannel rules communicate with the basic_nb and intra_nb notice boards. These rules monitor a number of failure and status bits in a specific channel. Whenever a bit is modified, the rule fires and calculates the total number of bits set in the group that the rule is monitoring.

The combining rule communicates with the basic_nb, intra_nb, and inter_nb notice boards and executes a unanimous voting operation across the three channels. It fires if it has three separate objects, in different channels, having the same name and value. It then obtains a handle on a resultant object having the same name but is in the inter channel nb. The value is then asserted into the resultant object.

The inter channel rules communicate with the basic_nb, intra_nb, and inter_nb notice boards. The inter channel rules are the first rules that operate on the combined output of each of the channels; that is, on data which has been generated by the combining rule. The results are placed in the inter_nb for further work in the inter channel ruleset and for the final stage of the analysis.

The inter multichannel rules communicate with the basic_nb, intra_nb, and inter_nb notice boards. These rules operate in the same manner as the intra multichannel rules. Instead of monitoring failure and status bits in one channel, however, they monitor bits for all three channels. Thus, if a bit is set in any channel that the rule is monitoring, the rule fires and recomputes the total number of bits set.

The FCS operability rules communicate with the basic_nb, intra_nb, inter_nb, and fcs_nb notice boards and are the final stage in the reasoning system. They take the output from the other rulesets and generate the system status indicators.

4.5 System Interface

The MUSE FSM system is interfaced to the Sun UNIX environment by means of the MUSE block sockets mechanism. This provides the capability for connecting any data transfer process desired to the input end of the sockets. The system is configured to use 12 MUSE data channels, with 4 for each FSM channel. A pair of data channels is used for each of the status and failure sections, one for setting bits on and the other for setting bits off. The interface checks that it is only propagating changes and, thus, will not notify the rule systems if the value received is the same as the existing value.

The support harness was written to allow the user the ability to set or unset failure or status bits in any of the three FSM channels. It also provides a facility to use command files which may contain set or unset commands, calls to other command files, comments, and print statements. This enables, for example, the setting of all the wheels down without having to issue individual weight on wheel commands for each of the three wheels in each of the three FSM channels. A facility to reset the MUSE system is also provided to eliminate the need to recompile the FSM code for each run.

4.6 Discussion of MUSE Implementation

Although many of the MUSE package's features were not tested by the FSM implementation, MUSE's flexibility in handling different knowledge representations was demonstrated. The variety of rulesets involved in the FSMKB were all successfully converted into MUSE structures.

The need to devise a special-purpose translator for conversion of the original LISP rules to MUSE format is symptomatic of a general comparison problem of the performance of different AI development environments. As yet, there is no standardized way of expressing the same knowledge in different systems. With small KBs, manual translation is feasible, but as the capability of knowledge-based systems improves, their complexity will inevitably increase and automatic translation will become essential. In general, the development of such translators demands a high level of skill and a considerable commitment of effort. On this program, for example, over 50 percent of the RAE effort was devoted to translator development.

In order to fully assess the performance of the prototype MUSE implementation of the FSMKB, further analysis of the initial conditions and of the representative failure modes is required. Because the MUSE rule system is only invoked when an input data item changes, there is no difficulty in maintaining real-time performance in this state. When the input data changes and the rule system is brought into play, however, the analysis time is generally less than 1 sec. This is short enough to be acceptable, but still longer than the time between successive data updates. A more detailed consideration of the foreground/background task management aspects of the MUSE prototype is needed to resolve this problem.

Preliminary performance data on a Sun 3/160 (2Mips) for MUSE compilation and initialization is as follows: For the full FSMKB, which comprises 400 plus rules and 400 input data items, on a Sun 3/160 (2Mips), the Retenet initialization takes 4–12 min. A revised formulation of the KB in MUSE to reduce the size of the Retenet is expected to reduce this overhead by at least an order of magnitude without significantly affecting run-time performance.

5 CONCLUDING REMARKS

The application of expert systems to flight test monitoring is particularly appropriate. The monitoring task is manpower and data intensive, and is well understood. The system's capabilities to monitor data downlinked from the flight test aircraft and to generate information on the state and health of the system for the monitoring engineers provides increased safety during flight testing of new systems. The expert system provides the systems engineers with ready access to the large amount of data required to describe a complex aircraft system; access to

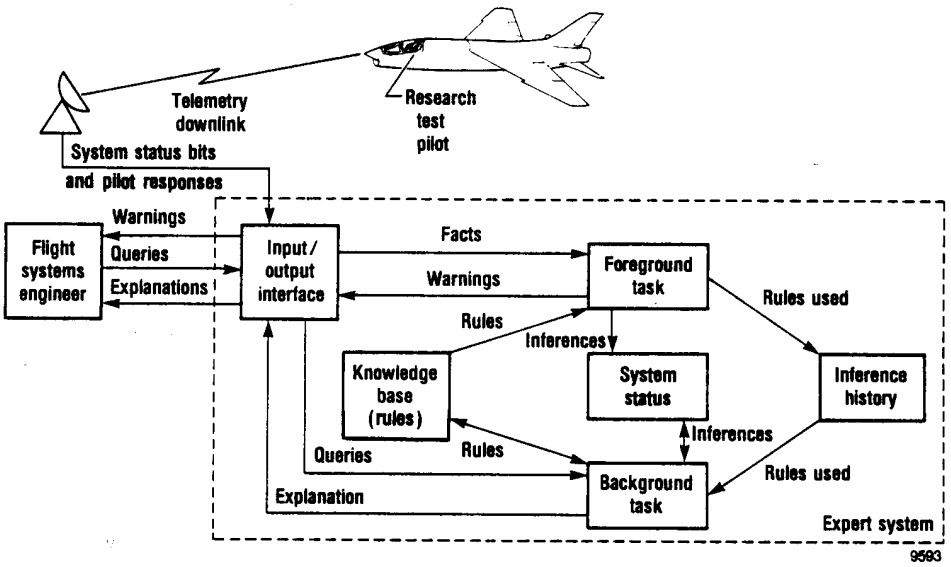

Fig. 1: Conceptual overview of flight status monitor structure.

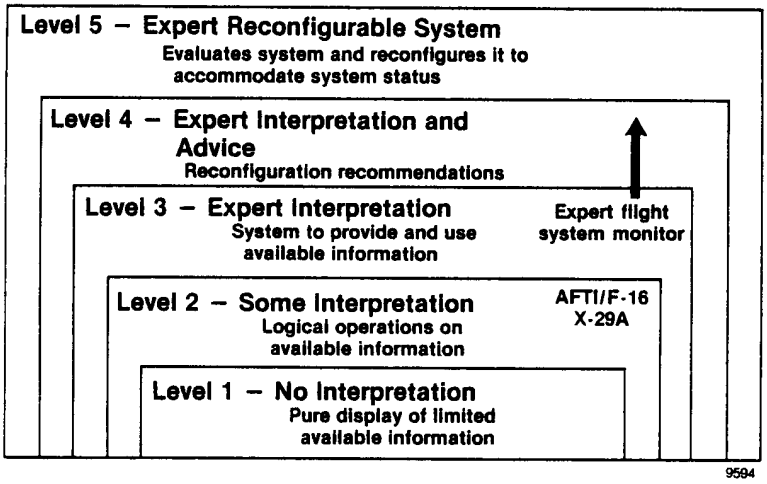

Fig. 2: Levels of information technology.

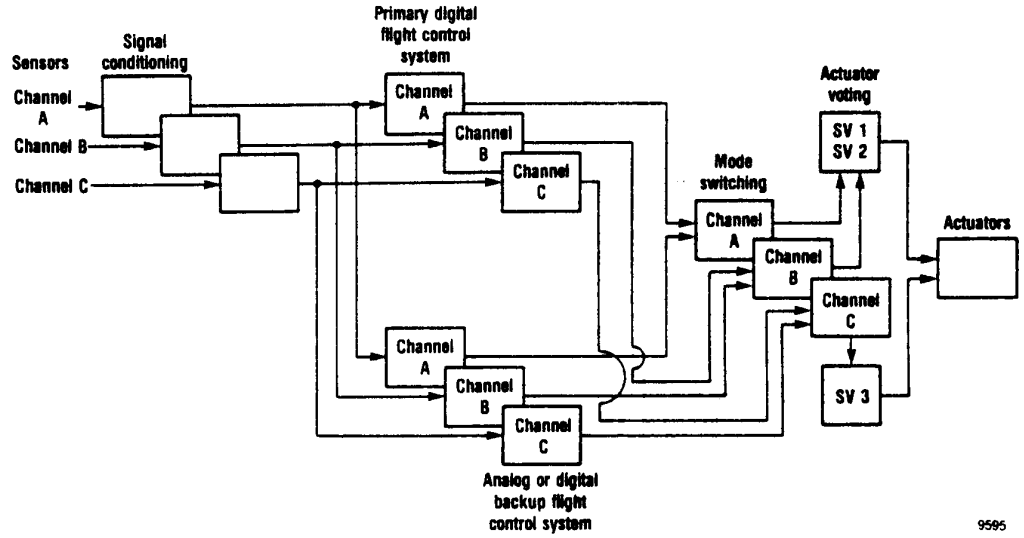

Fig. 3: Overview of representative digital flight control system.

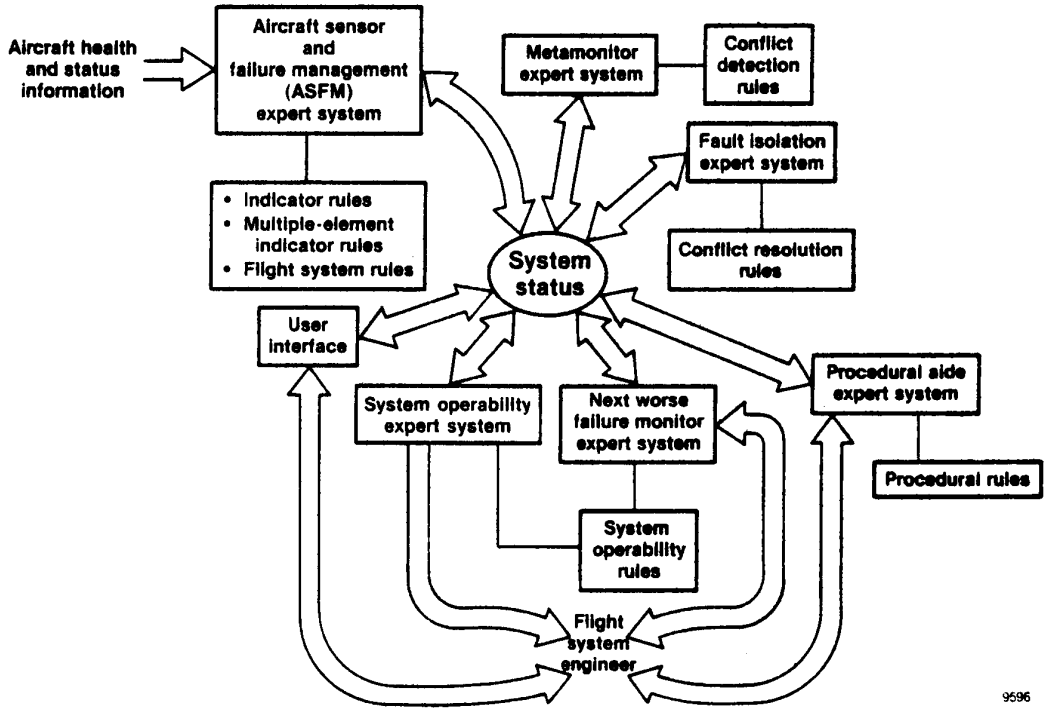

Fig. 4: Internal structure of flight status monitor.

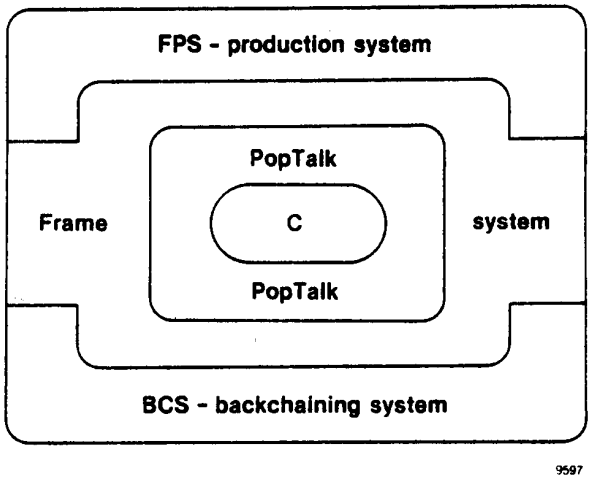

Fig. 5: MUSE knowledge representation integration.

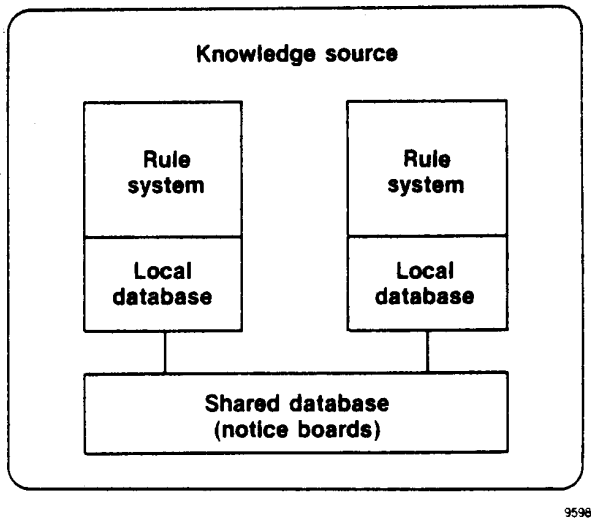

Fig. 6: MUSE knowledge source.

Original LISP rule:

```
(setq\f1 INTER-CHANNEL_RULE_8
        (make-Production_Rule
             :Name          "AR Condition"
             :Kind          "Conjunctive"
             :ANTECEDENTS '(
                       ("AR Mode Computed" "is" "off" )
                       ("Monitor Strakes" "is" "on" )
                                                        )
             :CONSEQUENTS '(
                       ("AR Condition" "is" "on" )
                       ("deduced AR Mode Indicator" "is" "off" )
                                                        )
             :Certainty  1.0
             :Explanation    "None given"
))
```

Translated MUSE rule:

```
/* Inter_Rule_8:  - AR Condition */
   || NASA explanation - None given
   ||
if
there is an item
        -tname "a_ar_mode_computed",
        -channel "i",
        -value 0
and
there is an item
        -tname "a_monitor_strakes",
        -channel "i",
        -value 1
and
there is an item I0,
        -tname "a_ar_condition",
        -channel "i"
and
there is an item I1,
        -tname "a_deduced_ar_mode_indicator",
        -channel "i"
then
assert {item I0: -value 1}
and
assert {item I1: -value 0}
and
do {
printf('Rule Inter_Rule_ 8 AR Condition has fired\');
}
```

Fig. 7: Comparison of FSM and MUSE rule representations.

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-101710		

4. Title and Subtitle

A Knowledge-Based Flight Status Monitor for Real-Time Application in Digital Avionics Systems

5. Report Date

October 1989

6. Performing Organization Code

7. Author(s)

E.L. Duke (NASA Ames Research Center); J.D. Disbrow (PRC Systems, Inc.); G.F. Butler (Royal Aerospace Establishment)

8. Performing Organization Report No.

H-1568

10. Work Unit No.

RTOP 505-66-71

9. Performing Organization Name and Address

NASA Ames Research Center
Dryden Flight Research Facility
P.O. Box 273, Edwards, CA 93523-5000

11. Contract or Grant No.

13. Type of Report and Period Covered

Technical Memorandum

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, DC 20546

14. Sponsoring Agency Code

15. Supplementary Notes

Prepared for presentation at the MILCOMP '89 Conference, September 26–28, 1989, at London, England.

16. Abstract

The Dryden Flight Research Facility of the National Aeronautics and Space Administration (NASA) Ames Research Center (Ames-Dryden) is the principal NASA facility for the flight testing and evaluation of new and complex avionics systems. To aid in the interpretation of system health and status data, a knowledge-based flight status monitor has been designed. The monitor is designed to use fault indicators from the onboard system which are telemetered to the ground and processed by a rule-based model of the aircraft failure management system to give timely advice and recommendations in the mission control room. One of the important constraints on the flight status monitor is the need to operate in real time, and to pursue this aspect, a joint research activity between NASA Ames-Dryden and the Royal Aerospace Establishment (RAE) on real-time knowledge-based systems has been established. Under this agreement, the original LISP knowledge base for the flight status monitor has been reimplemented using the intelligent knowledge-based system toolkit, MUSE, which was developed under RAE sponsorship. In this paper, details of the flight status monitor and the MUSE implementation are presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Aircraft systems Artificial intelligence Avionics Expert systems	Unclassified — Unlimited Subject category 63

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	14	A02

NASA FORM 1626 OCT 86

**For sale by the National Technical Information Service, Springfield, VA 22161-2171.*